UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
PRIMER SEMESTRE 2019
ING. OTTO ESCOBAR
TUTOR ACADÉMICO SECCIÓN A: RICARDO MENCHÚ
TUTOR ACADÉMICO SECCIÓN B: OSCAR CUELLAR

PROYECTO 2

Objetivo General:

- Aplicar los conocimientos adquiridos en el curso sobre el lenguaje ensamblador.

Objetivos Específicos:

- Aplicar el conocimiento de operaciones básicas y avanzadas a nivel ensamblador.
- Conocer el funcionamiento de las interrupciones.
- Comprender el uso de la memoria en los programas informáticos.
- Consolidar el uso del modo video en lenguaje a bajo nivel.
- Emplear y manipular la paleta de colores del modo video.
- Realizar todas las operaciones aritméticas a bajo nivel.

Descripción:

El proyecto final consiste en cargar un archivo que contendrá la estructura de un árbol genealógico, el cual tendrá la referencia a una imagen de cada miembro del árbol, a estas imágenes será posible aplicarles distintos tipos de filtros, las imágenes serán en formato (.bmp). Pueden realizar sus imágenes bmp en Paint.

El programa tendrá las siguientes opciones al inicio:

- **Menú Principal:**
  El menú principal es lo primero que se muestra al usuario, el cual contiene el encabezado con los datos mostrados a continuación y una serie de opciones las cuales son:

  - Cargar Archivo
  - Generar Árbol
  - Aplicar Filtros
  - Salir

  Encabezado:

  UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
  FACULTAD DE INGENIERÍA
  ESCUELA DE CIENCIAS Y SISTEMAS
  ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
  SECCIÓN A|B
  NOMBRE: OSCAR RENE CUELLAR MANCILLA
  CARNET: 201503712
  PROYECTO 2

- **Menú Cargar Archivo**

  Al seleccionar esta opción en el menú principal, el programa solicitará el ingreso de la ruta de un archivo de texto con extensión ".json", el programa debe validar que el archivo existe, en caso de no existir el archivo o poseer una extensión incorrecta se debe preguntar nuevamente la ruta. La ruta se ingresará con el siguiente formato:

  - **Carnet Pares**
    %\ruta\entrada.js%
    %entrada.json%
  - **Carnet Impares**
    &\ruta\entrada.js&
    &entrada.json&

  Se guardará la información necesaria para poder ser procesada luego.

- **Estructura JSON**

```
{
	"arbol":[
		{
			"oscar":{
				"image":"oscar.bmp",
				"idPareja":"0.1"
			}
		},
		{
			"susie":{
				"image":"susie.bmp",
				"idPareja":"+ 0.0 0.1"
			}
		},
		{
			"sofia":{
				"image":"sofia.bmp",
				"idHijo":"- 0.2 0.1"
			}
		},
		{
			"guadalupe":{
				"image":"guadalupe.bmp",
				"idHijo":"0.1",
				"idPareja":"1"
			}
		},
		{
			"pablo":{
				"image":"pablo.bmp",
				"idPareja":"1"
			}
		},
		{
			"allisson":{
				"image":"allisson.bmp",
				"idHijo":"0.1"
			}
		},
		{
			"andrea":{
				"image":"andrea.bmp",
				"idPareja":"/ 0.5 0.25"
			}
		},
		{
			"roberto":{
				"image":"roberto.bmp",
				"idPareja":"2"
			}
		}
	]
}
```

La estructura del archivo JSON tendrá únicamente un arreglo de objetos “miembros”, los cuales contienen 3 atributos:

- “image” (obligatorio) : contiene la referencia a una imagen que representa al miembro.
- “idPareja” (opcional): contiene un número que hace referencia al idPareja de otro miembro con el cual hacen pareja dentro del árbol.
- “idHijo” (opcional): contiene un número que hace referencia al idPareja de otro miembro que es su “padre”.

Aclaraciones:

- Los “idPareja”, “idHijo”, pueden tener directamente un número ó pueden tener una operación aritmética con operadores básicos (+,*,-,/).
- De tener una operación aritmética, estas vendrán en notación prefija. Ejemplo: + - + + 02 * 03 04 / 10 02 * 03 04 10.

- Pueden venir números con decimales.
- Pueden haber resultados decimales.
- No hay un límite de hijos.
- No hay un límite de parejas.

- **Aplicar Filtros**

Se tendrá una pequeña consola que únicamente recibirá 2 comandos “mostrar id”. Se podrán aplicar filtros a las imágenes dando como referencia el id del miembro, seguidamente se le podrán aplicar filtros y “detalles id”. Se podrán visualizar los detalles de la imagen. Ejemplo:

teniendo el miembro
“usac”:{
    “image”:”/logo.bmp”
}

Detalles:
>> detalles usac

Salida:
>> Ancho #, Alto #, Tamaño #

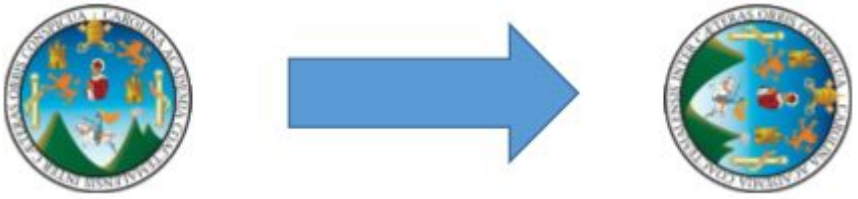
Ancho de la imagen: <ancho en pixeles>
Alto de la imagen: <alto en pixeles>
Tamaño de la imagen: <tamaño en bytes>

Mostrar:
>> mostrar usac

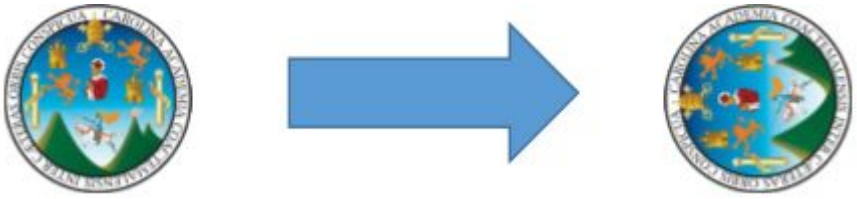
Esto cargará la imagen “/logo.bmp”, se mostrará en modo video. Al presionar la tecla ESC saldrá de mostrar la imagen. Luego de cargar la imagen, quedará mostrandose y esperando a que se le apliquen filtros.

Girar:
Teniendo la imagen mostrada se le podrá aplicar lo siguiente:

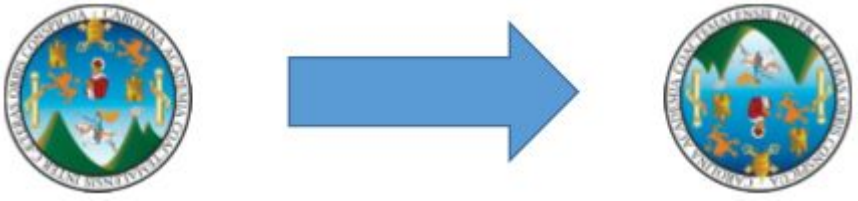
tecla flecha derecha (girará la imagen 90 grados a la derecha):



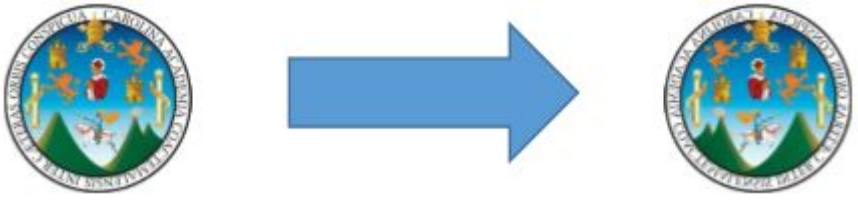
tecla flecha izquierda (girará la imagen 90 grados a la izquierda):



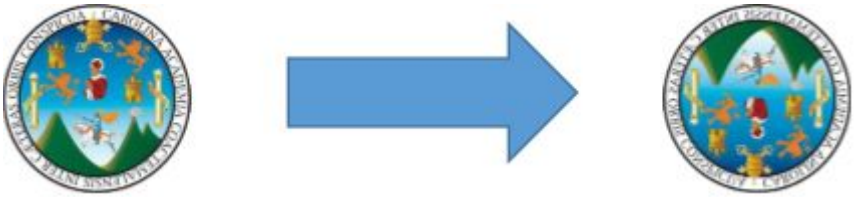
tecla flecha arriba (girará la imagen 180 grados):



tecla ctrl (voltear la imagen horizontalmente):

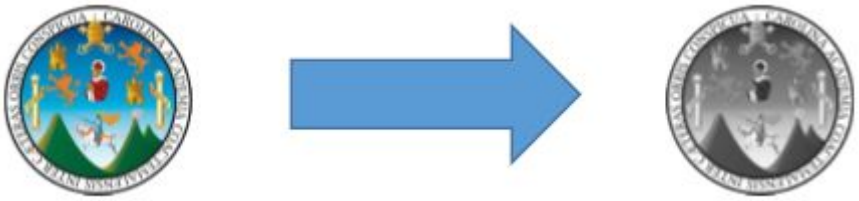


tecla shift (voltear la imagen verticalmente):

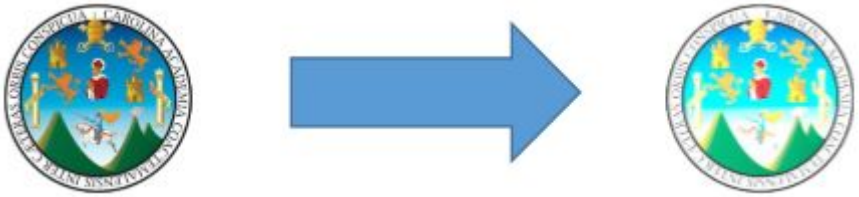


Filtros:

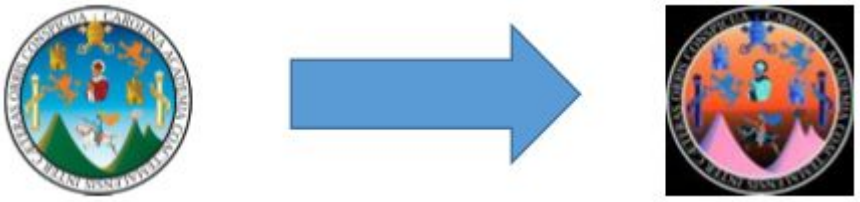
tecla G (establecerá la imagen en escala de grises):



tecla B (establecerá la imagen con brillo, la cantidad de brillo a aplicar queda a discreción del estudiante):



tecla N (establecerá la imagen con los colores invertidos, para aplicar el efecto negativo):



tecla S (guardará la imagen filtrada, reemplazará la original).

Aclaraciones:

- Tomar en cuenta que las imágenes que se usarán serán de color.
- Todas las imágenes serán con formato bmp.
- Todos los filtros son acumulables, es decir, puedo aplicar un filtro negativo a una imagen rotada 180 grados etc.

- **Generar Árbol**

  El programa será capaz de generar un archivo .dot, el cual se compila fuera del programa para generar una gráfica que muestre cómo está distribuido el árbol actualmente, el árbol que se graficara estará dado por el id del objeto JSON, se tendrá la opción de comandos ,como se muestra a continuación.

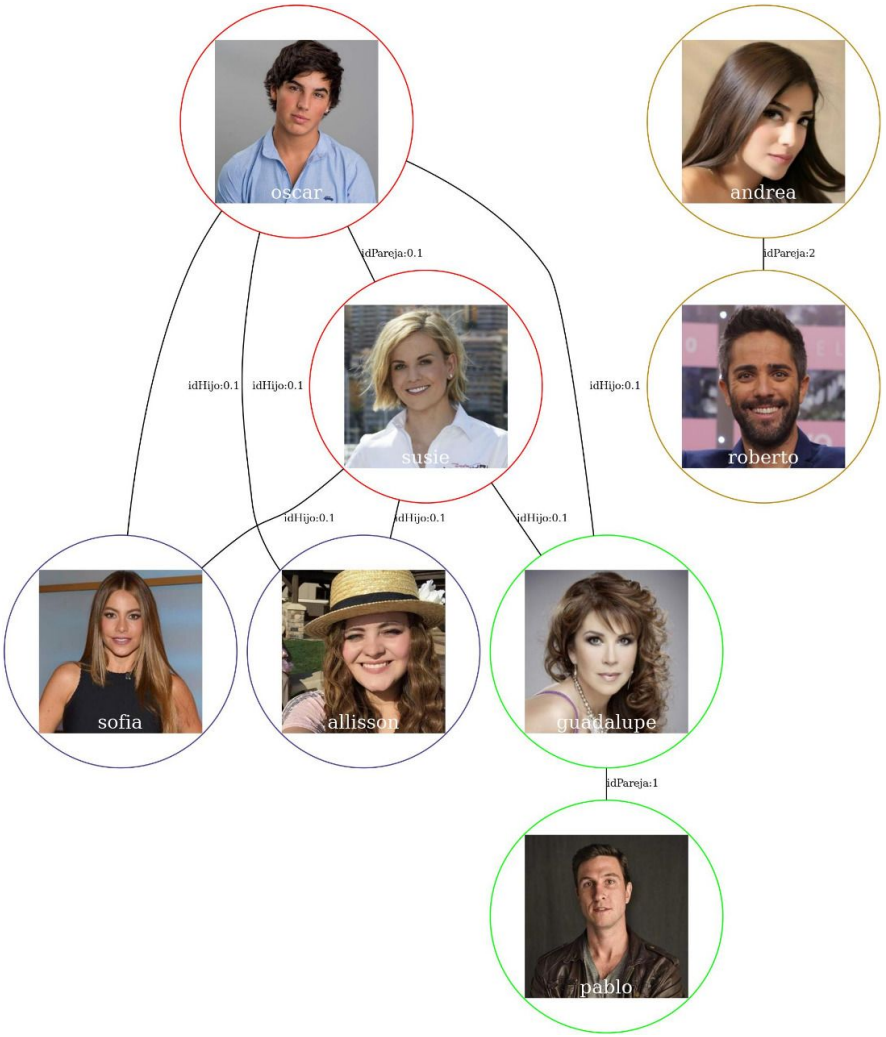
  >> graficar arbol

  “arbol” hace referencia al archivo padre de toda la estructura por lo cual genera el árbol de todos los miembros que hay dentro de él.

  Código Graphviz del ejemplo:

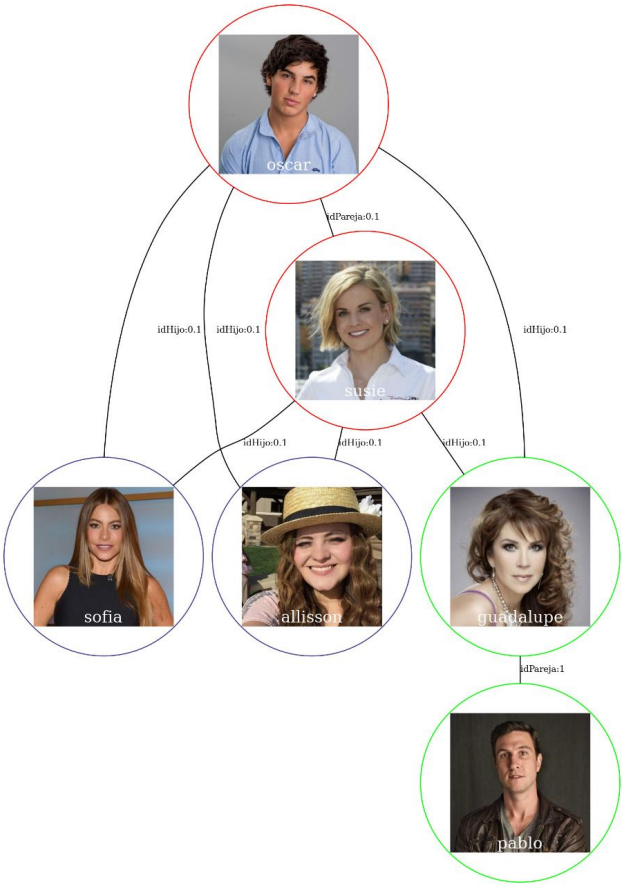
```
graph familia{
    node [fontsize=32.0,style=bold, labelloc=b, fontcolor=white];
    edge [style=bold, fontsize=18];
    oscar [color=red,label="oscar",image="oscar.bmp"];
    susie [color=red,label="susie",image="susie.bmp"];
    sofia [color=darkslateblue,label="sofia",image="sofia.bmp"];
    guadalupe [color=green,label="guadalupe",image="guadalupe.bmp"];
    pablo [color=green,label="pablo",image="pablo.bmp"];
    allisson [color=darkslateblue,label="allisson",image="allisson.bmp"];
    andrea [color=darkgoldenrod,label="andrea",image="andrea.bmp"];
    roberto [color=darkgoldenrod,label="roberto",image="roberto.bmp"];

    oscar -- susie [label="idPareja:0.1"];
    oscar -- sofia [label="idHijo:0.1"];
    susie -- sofia [label="idHijo:0.1"];
    oscar -- guadalupe [label="idHijo:0.1"];
    susie -- guadalupe [label="idHijo:0.1"];
    oscar -- allisson [label="idHijo:0.1"];
    susie -- allisson [label="idHijo:0.1"];
    guadalupe -- pablo [label="idPareja:1"];
    andrea -- roberto [label="idPareja:2"];
}
```

Imagen generada:



Aclaraciones:

- Cada uno de los miembros tendrá un nombre único.
- Como se muestra en la imagen, todas las relaciones deberán llevar el “idHijo” ó el “idPareja” con su respectivo número, ya operado. Ejemplo: “idHijo”:”1.0+1.1” debe mostrarse como idHijo:2.1, esto para tener referencia de haber operado bien los números.
- Al igual que se graficó “arbol” también puede graficarse cualquiera de sus miembros, por ejemplo:
  >> graficar oscar
  graficaría lo que se muestra a excepción de andrea y roberto, ya que no pertenecen al subárbol de oscar. ejemplo:



- **Salir**

  Al seleccionar esta última opción el programa finalizará y saldrá al sistema de DosBox.

Referencias

Graphviz:
https://www.graphviz.org/gallery/

Observaciones y Restricciones:

- **Se realizará de manera individual.**
- **Copias totales o parciales tendrán una nota de 0 y serán reportadas a escuela.**
- El código del programa debe ser estrictamente ensamblador, no se permite el uso de alguna librería.
- El entorno de pruebas a utilizar debe ser DOSBox, el ensamblador a utilizar queda a discreción del estudiante, por ejemplo: MASM, NASM, TASM, FASM, etc.
- El día de la calificación se harán preguntas sobre aspectos utilizados en la elaboración del proyecto, las cuales se considerarán en la nota final.
- Utilizar el modo de video 13h.

Requerimientos Mínimos

- Para tener derecho a calificación:
    - Se debe presentar el proyecto en DOSBox.
    - Se debe haber entregado manual de usuario y manual técnico, de lo contrario se asumirá que el estudiante copió.
    - Lectura de archivo.
    - Filtro de imágenes (ver la imagen cargada, girar la imagen en 180° y aplicar el filtro de escala de grises, siempre utilizando el modo de video 13h.).

- Enviar archivo con el código utilizado para la práctica y manual técnico antes de las 21:59 horas del Lunes 29 de Abril de 2019.
  ➢ nombre: [ARQ1]P2_#Carnet.rar
  ➢ medio: Classroom.

Fecha de Calificación:
Martes 30 de Abril, el horario y lugar se informará en los días próximos a la fecha de entrega.
SIN PRÓRROGA.